update-project-3

April 12, 2025

[71]:

```
import pandas as pd
import numpy as np
# Load dataset from a CSV file
df2 = pd.read_csv('data 2.csv')
df1=pd.read_csv('data 5.csv')
#df3=pd.read_csv('new data3.csv')
#df3=pd.read_csv('project data only.csv')
df3=pd.read_csv('new datak.csv')
```

[72]:

```
df2.head()
```

[72]:

```
   User_Id  Place_Id  Place_rating
0        5         1           4.1
1       40         2           4.2
2    11799         3           4.6
3       81         4           3.1
4       69         5           3.7
```

[73]:

```
df1.head()
```

[73]:

```
   Place_Id       Source  Destination  Distance(km)
0         1       Amtala    Bishnupur           2.2
1         2    Bishnupur   Khoriberia           4.6
2         3   Khoriberia  Vasa Mandir           1.5
3         4  Vasa mandir       Pailan           5.4
4         5       Pailan         Joka           5.0
```

[74]:

```
df1.count
```

[74]:

```
<bound method DataFrame.count of      Place_Id                  Source
Destination  Distance(km)
0           1                     Amtala          Bishnupur          2.20
1           2                  Bishnupur         Khoriberia          4.60
2           3                 Khoriberia        Vasa Mandir          1.50
3           4                Vasa mandir             Pailan          5.40
4           5                     Pailan               Joka          5.00
..        …                          …                …           …
```

```
165        166  BK Pal(Rabindra Sarani)          Ahiritola          1.20
166        167                Ahiritola         Jora Bagan          0.85
167        168               Jora Bagan          Mala para          0.21
168        169                Mala para  Satyanarayan Park          1.30
169        170        Satyanarayan Park          Barabazar          0.50

[170 rows x 4 columns]>
```

[75]:
```
df2.count
```

[75]:
```
<bound method DataFrame.count of      User_Id  Place_Id  Place_rating
0          5         1          4.1
1         40         2          4.2
2      11799         3          4.6
3         81         4          3.1
4         69         5          3.7
..        …       …           …
165        6       166          4.3
166        9       167          3.8
167       48       168          3.9
168    18154       169          4.1
169        4       170          3.5

[170 rows x 3 columns]>
```

[76]:
```
df3.head()
```

[76]:
```
   Place_Id            Place_name       Age  \
0         1     Victoria Memorial  All Ages
1         2            Quest Mall  All Ages
2         3  Fort William Kolkata  All Ages
3         4      Shalimar Station  All Ages
4         5            Belur Math  All Ages

                               Category Road_condition Weather_Condition  \
0                   Historical Monument           Good              Haze
1               Shopping, Entertainment           Good              Haze
2                       Historical Site           Good            Cloudy
3  Transportation Hub (Railway Station)           Good              Hazr
4                   Religious/Spiritual           Good              Haze

                                         Description Mode_of_Transport  \
0  A grand white marble monument dedicated to Que…          Bus, Taxi
1  A modern, upscale shopping mall with various b…   Bus, Taxi, Metro
2  A historic British fort with a museum showcasi…          Bus, Taxi
3  A major railway station in Howrah, Kolkata, se…   Train, Bus, taxi
4  The headquarters of the Ramakrishna Mission, a…   Bus, Taxi, Ferry
```

```
   Latitude  Longitude
0  22.54498   88.34243
1  22.53915   88.36603
2  22.55895   88.33773
3  22.55591   88.31503
4  22.63282   88.35642
```

[77]:

```
df3.count
```

[77]:

```
<bound method DataFrame.count of      Place_Id
Place_name       Age  \
0          1                              Victoria Memorial  All Ages
1          2                                     Quest Mall  All Ages
2          3                           Fort William Kolkata  All Ages
3          4                               Shalimar Station  All Ages
4          5                                     Belur Math  All Ages
..        …                                            …       …
165       166  Sabuj Sathi Krirangan (Howrah Indoor Stadium)  All Ages
166       167                                 Behala Airport  All Ages
167       168                        Metropolitan Durga Bari  All Ages
168       169                                     Atmosphere  All Ages
169       170            Abanindranath Tagore's Garden House  All Ages

                                           Category Road_condition  \
0                               Historical Monument           Good
1                           Shopping, Entertainment           Good
2                                   Historical Site           Good
3              Transportation Hub (Railway Station)           Good
4                               Religious/Spiritual           Good
..                                              …            …
165                    Indoor Stadium, Sports Venue        Average
166                      Airport, Aviation Training        Average
167                          Temple, Religious Site        Average
168                                 floating bridge        Average
169  Heritage House, Garden, Art, Museum (Possible)        Average

    Weather_Condition                                        Description  \
0                Haze  A grand white marble monument dedicated to Que…
1                Haze  A modern, upscale shopping mall with various b…
2              Cloudy  A historic British fort with a museum showcasi…
3                Hazr  A major railway station in Howrah, Kolkata, se…
4                Haze  The headquarters of the Ramakrishna Mission, a…
..                 …                                                …
165             Clear  An indoor stadium in Howrah, West Bengal, used…
166              Haze  A small airport in Kolkata primarily used for …
167              Haze  A Hindu temple dedicated to Goddess Durga, kno…
```

```
168             Haze  It is a sky bridge, called Deya, is the world'…
169             Haze  The former residence and garden of the renowne…

                        Mode_of_Transport  Latitude  Longitude
0                              Bus, Taxi  22.54498   88.34243
1                       Bus, Taxi, Metro  22.53915   88.36603
2                              Bus, Taxi  22.55895   88.33773
3                       Train, Bus, taxi  22.55591   88.31503
4                       Bus, Taxi, Ferry  22.63282   88.35642
..                                    …         …          …
165  Bus, Taxi , Auto-rickshaw, Train  22.58170   88.30680
166         Bus, Taxi , Auto-rickshaw  22.50400   88.29430
167         Bus, Taxi , Auto-rickshaw  22.54000   88.40800
168         Bus, Taxi , Auto-rickshaw  22.63792   88.45364
169         Bus, Taxi , Auto-rickshaw  22.70510   88.34450

[170 rows x 10 columns]>
```

[78]:
```
df3.columns
```

[78]:
```
Index(['Place_Id', 'Place_name', 'Age', 'Category', 'Road_condition',
       'Weather_Condition', 'Description', 'Mode_of_Transport', 'Latitude',
       'Longitude'],
      dtype='object')
```

[ ]:
```

```

[79]:
```
# Merge ratings with place info
df = pd.merge(df2, df3, on='Place_Id', how='left')
```

[80]:
```
df
```

[80]:
```
     User_Id  Place_Id  Place_rating  \
0          5         1           4.1
1         40         2           4.2
2      11799         3           4.6
3         81         4           3.1
4         69         5           3.7
..         …         …             …
165        6       166           4.3
166        9       167           3.8
167       48       168           3.9
168    18154       169           4.1
169        4       170           3.5

                                  Place_name       Age  \
0                          Victoria Memorial  All Ages
```

```
1                                       Quest Mall  All Ages
2                             Fort William Kolkata  All Ages
3                                 Shalimar Station  All Ages
4                                       Belur Math  All Ages
..                                              …       …
165  Sabuj Sathi Krirangan (Howrah Indoor Stadium)  All Ages
166                                 Behala Airport  All Ages
167                        Metropolitan Durga Bari  All Ages
168                                     Atmosphere  All Ages
169            Abanindranath Tagore's Garden House  All Ages

                                           Category Road_condition  \
0                               Historical Monument           Good
1                           Shopping, Entertainment           Good
2                                   Historical Site           Good
3              Transportation Hub (Railway Station)           Good
4                               Religious/Spiritual           Good
..                                               …            …
165                    Indoor Stadium, Sports Venue        Average
166                      Airport, Aviation Training        Average
167                          Temple, Religious Site        Average
168                                 floating bridge        Average
169  Heritage House, Garden, Art, Museum (Possible)        Average

    Weather_Condition                                        Description  \
0                Haze  A grand white marble monument dedicated to Que…
1                Haze  A modern, upscale shopping mall with various b…
2              Cloudy  A historic British fort with a museum showcasi…
3                Hazr  A major railway station in Howrah, Kolkata, se…
4                Haze  The headquarters of the Ramakrishna Mission, a…
..                 …                                                …
165             Clear  An indoor stadium in Howrah, West Bengal, used…
166              Haze  A small airport in Kolkata primarily used for …
167              Haze  A Hindu temple dedicated to Goddess Durga, kno…
168              Haze  It is a sky bridge, called Deya, is the world'…
169              Haze  The former residence and garden of the renowne…

                    Mode_of_Transport  Latitude  Longitude
0                           Bus, Taxi  22.54498   88.34243
1                    Bus, Taxi, Metro  22.53915   88.36603
2                           Bus, Taxi  22.55895   88.33773
3                    Train, Bus, taxi  22.55591   88.31503
4                    Bus, Taxi, Ferry  22.63282   88.35642
..                                  …       …         …
165  Bus, Taxi , Auto-rickshaw, Train  22.58170   88.30680
166         Bus, Taxi , Auto-rickshaw  22.50400   88.29430
167         Bus, Taxi , Auto-rickshaw  22.54000   88.40800
```

```
168         Bus, Taxi , Auto-rickshaw  22.63792   88.45364
169         Bus, Taxi , Auto-rickshaw  22.70510   88.34450

[170 rows x 12 columns]
```

[81]:

```
df.columns
```

[81]:

```
Index(['User_Id', 'Place_Id', 'Place_rating', 'Place_name', 'Age', 'Category',
       'Road_condition', 'Weather_Condition', 'Description',
       'Mode_of_Transport', 'Latitude', 'Longitude'],
      dtype='object')
```

[ ]:

[ ]:

[82]:

```
# Merge ratings with place info
df = pd.merge(df2, df3, on='Place_Id', how='left')

# Check and drop nulls if necessary
df = df.dropna()



#Create the encoded dataframe
df_encoded = pd.get_dummies(df[['Place_name', 'Age', 'Category',␣
 ↪'Mode_of_Transport']], drop_first=True)

# Add rating as a feature
df_encoded['Place_rating'] = df['Place_rating']
```

[ ]:

[83]:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded)
```

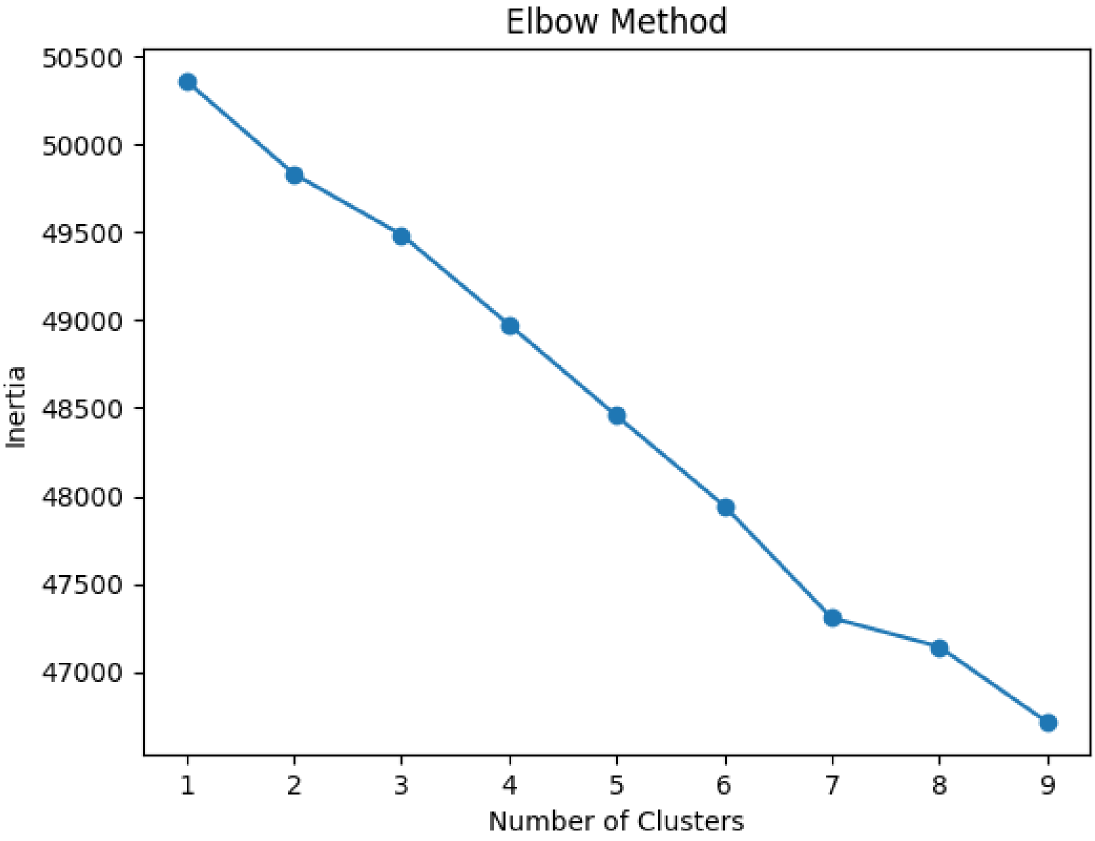
[84]:

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Elbow method to find optimal K
sse = []
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    sse.append(kmeans.inertia_)
```

```python
plt.plot(range(1, 10), sse, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()

# Based on elbow curve, suppose we choose k=4
kmeans = KMeans(n_clusters=4, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
```



[85]:

```python
# Example: Recommend places similar to a given place
target_place = "Howrah Bridge"
target_cluster = df[df['Place_name'] == target_place]['Cluster'].values[0]

# Get all places in the same cluster (excluding the selected one)
recommendations = df[(df['Cluster'] == target_cluster) & (df['Place_name'] !=␣
 ↪target_place)]

# Show unique recommendations
```

```python
print("Recommended places similar to", target_place)
print(recommendations[['Place_name', 'Age', 'Category', 'Mode_of_Transport']].
  ↪drop_duplicates())
```

```
Recommended places similar to Howrah Bridge
                                        Place_name       Age  \
0                                Victoria Memorial  All Ages
1                                       Quest Mall  All Ages
2                             Fort William Kolkata  All Ages
3                                 Shalimar Station  All Ages
4                                       Belur Math  All Ages
..                                             …       …
165  Sabuj Sathi Krirangan (Howrah Indoor Stadium)  All Ages
166                                 Behala Airport  All Ages
167                        Metropolitan Durga Bari  All Ages
168                                     Atmosphere  All Ages
169            Abanindranath Tagore's Garden House  All Ages

                                           Category  \
0                               Historical Monument
1                           Shopping, Entertainment
2                                   Historical Site
3              Transportation Hub (Railway Station)
4                               Religious/Spiritual
..                                                …
165                    Indoor Stadium, Sports Venue
166                     Airport, Aviation Training
167                          Temple, Religious Site
168                                 floating bridge
169  Heritage House, Garden, Art, Museum (Possible)

                    Mode_of_Transport
0                           Bus, Taxi
1                    Bus, Taxi, Metro
2                           Bus, Taxi
3                    Train, Bus, taxi
4                    Bus, Taxi, Ferry
..                                  …
165  Bus, Taxi , Auto-rickshaw, Train
166         Bus, Taxi , Auto-rickshaw
167         Bus, Taxi , Auto-rickshaw
168         Bus, Taxi , Auto-rickshaw
169         Bus, Taxi , Auto-rickshaw

[165 rows x 4 columns]
```

[86]:

```python
print(df['Place_name'].unique())
```

```
['Victoria Memorial' 'Quest Mall' 'Fort William Kolkata'
 'Shalimar Station' 'Belur Math' 'Howrah Bridge' 'Birla Planetarium'
 'Indian Museum' 'Marin House' 'Marble Palace Mansion' 'Mother House'
 'Science City Kolkata' "St. Paul's Cathedral Kolkata" 'Tea Board'
 'Tajpur' 'Birla Mandir Kolkata' 'Eden Gardens' 'Jorasanko Thakur Bari'
 'Birla Industrial & Technological Museum' 'Rabindra Sarovar'
 'Kalighat Temple' 'Shobhabajar Rajbari' 'Botanical Garden in Kolkata'
 'Nakhoda Mosque' 'Alipore Zoo' 'Sabarna Sangrahashala' 'Eco Tourism Park'
 'Calcutta Jain Temple' 'Nicco Park' 'Prinsep Ghat' 'Aquatica'
 'Park Street' 'Chowringhee' 'ISKCON Kolkata' 'South Park Street Cemetery'
 'Netaji Bhawan' "St John's Church" 'Barrackpore'
 'Sabarna Roy Chowdhury Sangrahashala' "Nehru Children's Museum"
 'The RBI Museum' 'Smaranika Tram Museum' 'Maulana Azad Museum' 'Maidan'
 'Central Park' 'Millenium Park' 'Deshapriya Park' 'Safari Park'
 'Mohor Kunja' 'Elliot Park' 'Gitanjali Sports Complex'
 'Kishore Bharati Krirangan' 'Salt Lake Stadium'
 'Rabindra Sarobar Stadium' 'Mohunbagan Stadium' 'Netaji Indoor Stadium'
 'Barasat Stadium' 'East Bengal Ground' 'Nandan' 'Nalban Boating Park'
 'Snow Park' 'Wet-O-Wild' 'Genesis Art Gallery' 'Galerie 88'
 'Experimenter Art Gallery' 'Masters Collection Art Gallery'
 'Chitrakoot Art Gallery' 'Akar Prakar Gallery' 'Aakriti Art Gallery'
 'Chemould Art Gallery' 'Janus Art Gallery'
 'Harrington Street Arts Centre' 'CIMA Gallery Pvt Ltd'
 'Magen David Synagogue' 'Beth El Synagogue' 'Neveh Shalome Synagogue'
 'Dakshineswar Kali Mandir' 'Ramkrishnapur Ghat' 'Kolkata Police Museum'
 'Muktangan' 'Gariahat Market' 'Kolkata Port Trust' 'Gurusaday Museum'
 'Metcalfe Hall' 'Mullick Ghat Flower Market' 'College Street (Boi Para)'
 'College Square' 'Nahoum and Sons Bakery'
 'Nipponzan Myohoji Buddhist Temple' 'Chinatown (Tiretta Bazaar)'
 "Park Street's Iconic Eateries" 'Alipore Jail Museum'
 'Chintamoni Kar Bird Sanctuary' 'Boat Museum' 'Lal Dighi' 'Kumartuli'
 'Rail Museum' 'Ahuja Museum for Arts' 'Sovabazar Rajbari'
 'Sri Mahalakshmi Temple' 'Raja Rammohan Roy Memorial Museum'
 'Pareshnath Jain Temple' 'Baranagar Math' 'Ratan Babu Ghat'
 'Cossipore Gun & Shell Factory Museum' 'Shree Sachiyay Mata ji Mandir'
 'Salt Lake City Center (Bidhannagar)' 'Chinese Kali Temple'
 'Ecospace Business Park' 'Barasat Krishnamati Park'
 'Kestopur Rabindra Tirtha' 'Baghbazar Ghat' 'Cossipore Udyanbati'
 'Vivekananda Setu' 'Currency Building' 'South City Mall'
 'Tollygunge Golf Club' 'State Archaeological Museum' 'National Library'
 'Biswa Bangla Gate' 'Sundarbans National Park'
 'Alipore Zoological Garden' 'Kolkata Race Course'
 'Achipur Chinese Temple' 'Falta River Side' 'Diamond Harbour River Side'
 'Raichak river side' 'Diamond harbour purano kella'
 'BAPS Shri Swaminarayan Mandir' 'Henry’s Island' 'Mani Square'
 'Shri RamChandra Mandir' 'Howrah Railway Station' 'Vidyasagar Setu'
 'Barisha Chandi Temple' 'Nature park' 'Shri Jagannath Temple'
 'Royal Calcutta Turf Club' 'Kala Mandir' 'Fancy Market' 'Entally Market'
 'Acropolis Mall' 'Eco Kunj' 'Kunjo Chhaya' 'Nazrul Tirtha'
 'Lake Kalibari' 'Shri Shri Karunamoyee Kali Mandir' 'Bhootnath Mandir'
 'Nataji Subhash Chandra Bose International Airport' 'Taj Bengal'
 'Dhono Dhanyo Auditorum' 'Doi Ghat' 'Aircraft Museum' 'Arts Acre'
 'B Garden' 'Patuli Jheel Park' 'Golf Green Central Park'
 'SECTOR - V  ViewPoint' 'Culture And Heritage of Bengal' 'Nature Park'
 "Chandraketu's Fort" 'Dhanyakuria Gayen Mansion'
 'Dhanyakuria Ballav Bari' 'Baropol Park Dhanyakuri'
 'Sabuj Sathi Krirangan (Howrah Indoor Stadium)' 'Behala Airport'
 'Metropolitan Durga Bari' 'Atmosphere'
 "Abanindranath Tagore's Garden House"]
```

[ ]:

[ ]:

[ ]:

[87]:

```
#!pip install geopy
```

[88]:

```
from geopy.distance import geodesic
```

[89]:

```
df.columns
```

[89]:

```
Index(['User_Id', 'Place_Id', 'Place_rating', 'Place_name', 'Age', 'Category',
       'Road_condition', 'Weather_Condition', 'Description',
       'Mode_of_Transport', 'Latitude', 'Longitude', 'Cluster'],
      dtype='object')
```

[90]:

```
# Optional: clean column names
df.columns = df.columns.str.strip()

# Drop rows with missing coordinates or ratings
df.dropna(subset=['Latitude', 'Longitude', 'Place_rating'], inplace=True)
```

[91]:

```
# Use features for clustering
features = df[['Latitude', 'Longitude', 'Place_rating']]
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Apply KMeans clustering
kmeans = KMeans(n_clusters=5, random_state=42)
```

[ ]:

[92]:

```
# User input
target_place = input("Enter the place you want to visit: ").strip()
```

```
# Check if it exists
if target_place not in df['Place_name'].values:
    print(f"'{target_place}' not found in the dataset.")
else:
    # Get target place details
    target_data = df[df['Place_name'] == target_place].iloc[0]
    target_cluster = target_data['Cluster']
    target_location = (target_data['Latitude'], target_data['Longitude'])

    # Recommend places in the same cluster, excluding the selected one
    recommendations = df[(df['Cluster'] == target_cluster) & (df['Place_name'] !
 ↪= target_place)].copy()

    # Calculate distances
    recommendations['Distance_km'] = recommendations.apply(
        lambda row: geodesic(target_location, (row['Latitude'],␣
 ↪row['Longitude'])).km,
        axis=1
    )

    # Suggest mode of transport
    def suggest_mode(dist):
        if dist < 1:
            return 'Walk'
        elif dist < 5:
            return 'Auto/Rickshaw'
        elif dist < 20:
            return 'Cab'
        else:
            return 'Metro/Bus'

    recommendations['Suggested_Transport'] = recommendations['Distance_km'].
 ↪apply(suggest_mode)

    # Sort by nearest
    recommendations = recommendations.sort_values('Distance_km')

    # Display top 5 recommendations
    print("\nTop nearby recommendations:")
    print(recommendations[['Place_name', 'Age', 'Category', 'Distance_km',␣
 ↪'Mode_of_Transport']].head())
```

```
Enter the place you want to visit:  Botanical Garden in Kolkata


Top nearby recommendations:
                                       Place_name       Age   \
3                                   Shalimar Station  All Ages
165  Sabuj Sathi Krirangan (Howrah Indoor Stadium)  All Ages
160                                       Nature Park    Adults
139                                      Fancy Market  All Ages
136                             Shri Jagannath Temple  All Ages

                                 Category  Distance_km  \
3    Transportation Hub (Railway Station)     2.719953
165          Indoor Stadium, Sports Venue     2.813631
160                           nature view     2.847313
139                      Market, Shopping     3.423767
136                Temple, Religious Site     3.752161

                    Mode_of_Transport
3                    Train, Bus, taxi
165  Bus, Taxi , Auto-rickshaw, Train
160                         Bus, Taxi
139                 Bus, Taxi , Metro
136          Bus, Taxi, Auto-rickshaw
```

[53]:

```
# User input
target_place = input("Enter the place you want to visit: ").strip()

# Check if it exists
if target_place not in df['Place_name'].values:
    print(f"'{target_place}' not found in the dataset.")
else:
    # Get target place details
    target_data = df[df['Place_name'] == target_place].iloc[0]
    target_cluster = target_data['Cluster']
    target_location = (target_data['Latitude'], target_data['Longitude'])

    # Recommend places in the same cluster, excluding the selected one
    recommendations = df[(df['Cluster'] == target_cluster) & (df['Place_name'] !
 ↪= target_place)].copy()

    # Calculate distances
    recommendations['Distance_km'] = recommendations.apply(
        lambda row: geodesic(target_location, (row['Latitude'],␣
 ↪row['Longitude'])).km,
        axis=1
    )

    # Suggest mode of transport
    def suggest_mode(dist):
        if dist < 1:
            return 'Walk'
        elif dist < 5:
            return 'Auto/Rickshaw'
        elif dist < 20:
            return 'Cab'
        else:
            return 'Metro/Bus'

    recommendations['Suggested_Transport'] = recommendations['Distance_km'].
 ↪apply(suggest_mode)

    # Sort by nearest
    recommendations = recommendations.sort_values('Distance_km')

    # Display top 5 recommendations
    print("\nTop nearby recommendations:")
    print(recommendations[['Place_name', 'Age', 'Category', 'Distance_km',␣
 ↪'Mode_of_Transport']].head())
```

```
Enter the place you want to visit:  Fancy Market

Top nearby recommendations:
           Place_name              Age  \
133   Vidyasagar Setu         All Ages
3    Shalimar Station         All Ages
8         Marin House  Teens, Families
151          Doi Ghat         All Ages
29       Prinsep Ghat         All Ages

                                    Category  Distance_km  \
133                      Cable-stayed bridge     0.546225
3       Transportation Hub (Railway Station)     0.799134
8    Historical Mansion, Museum, Art Gallery     0.843218
151                     Ghat, Religious Site     0.883755
29                               Scenic Spot     1.058318

            Mode_of_Transport
133            Bus,Taxi,Train
3            Train, Bus, taxi
8    Bus, Taxi, Auto-rickshaw
151  Bus, Taxi, Auto-rickshaw
29           Bus, Taxi, Ferry
```

[93]:

```
# User input
target_place = input("Enter the place you want to visit: ").strip()

# Check if it exists
if target_place not in df['Place_name'].values:
    print(f"'{target_place}' not found in the dataset.")
else:
    # Get target place details
    target_data = df[df['Place_name'] == target_place].iloc[0]
    target_cluster = target_data['Cluster']
    target_location = (target_data['Latitude'], target_data['Longitude'])

    # Recommend places in the same cluster, excluding the selected one
    recommendations = df[(df['Cluster'] == target_cluster) & (df['Place_name'] !
 ↪= target_place)].copy()

    # Calculate distances
    recommendations['Distance_km'] = recommendations.apply(
        lambda row: geodesic(target_location, (row['Latitude'],␣
 ↪row['Longitude'])).km,
        axis=1
    )

    # Suggest mode of transport
    def suggest_mode(dist):
        if dist < 1:
            return 'Walk'
        elif dist < 5:
            return 'Auto/Rickshaw'
        elif dist < 20:
            return 'Cab'
        else:
            return 'Metro/Bus'

    recommendations['Suggested_Transport'] = recommendations['Distance_km'].
 ↪apply(suggest_mode)

    # Sort by nearest
    recommendations = recommendations.sort_values('Distance_km')

    # Display top 5 recommendations
    print("\nTop nearby recommendations:")
    print(recommendations[['Place_name', 'Age', 'Category', 'Distance_km',␣
 ↪'Mode_of_Transport']].head())
```

```
Enter the place you want to visit:  Diamond Harbour River Side


Top nearby recommendations:
                       Place_name       Age                 Category  \
127  Diamond harbour purano kella  All Ages          Riverside views
126            Raichak river side  All Ages          Riverside views
124              Falta River Side  All Ages                Riverside
```

```
123        Achipur Chinese Temple  All Ages  A historic Chinese temple
154                      B Garden  All Ages     Residential Area, Park

     Distance_km          Mode_of_Transport
127     2.482020             Bus,Taxi,Ferry
126     7.216923             Bus,Taxi,Train
124    14.070730             Bus,Taxi,Train
123    29.238913             Bus,Taxi,Metro
154    30.896919  Bus, Taxi , Auto-rickshaw
```

[ ]:

```

```

[ ]:

```

```

[95]:

```python
# User input
target_place = input("Enter the place you want to visit: ").strip()

# Check if it exists
if target_place not in df['Place_name'].values:
    print(f"'{target_place}' not found in the dataset.")
else:
    # Get target place details
    target_data = df[df['Place_name'] == target_place].iloc[0]
    target_cluster = target_data['Cluster']
    target_location = (target_data['Latitude'], target_data['Longitude'])

    # Recommend places in the same cluster, excluding the selected one
    recommendations = df[(df['Cluster'] == target_cluster) & (df['Place_name'] !
 ↪= target_place)].copy()

    # Calculate distances
    recommendations['Distance_km'] = recommendations.apply(
        lambda row: geodesic(target_location, (row['Latitude'],␣
 ↪row['Longitude'])).km,
        axis=1
    )

    # Suggest mode of transport
    def suggest_mode(dist):
        if dist < 1:
            return 'Walk'
        elif dist < 5:
            return 'Auto/Rickshaw'
        elif dist < 20:
            return 'Cab'
        else:
            return 'Metro/Bus'
    recommendations['Suggested_Transport'] = recommendations['Distance_km'].
  ↪apply(suggest_mode)

    # Sort by nearest
    recommendations = recommendations.sort_values('Distance_km')

    # Display top 5 recommendations
    print("\nTop nearby recommendations:")
    print(recommendations[['Place_name', 'Age', 'Category', 'Distance_km',␣
  ↪'Mode_of_Transport']].head())
```

```
Enter the place you want to visit:  College Street (Boi Para)


Top nearby recommendations:
                      Place_name       Age               Category  Distance_km  \
86                College Square  All Ages  Book market Park Area     0.178029
9          Marble Palace Mansion  All Ages     Historical Mansion     0.785765
23                Nakhoda Mosque  All Ages    Religious/Spiritual     0.833912
89    Chinatown (Tiretta Bazaar)  All Ages      Chinese heritage      0.881687
131       Shri RamChandra Mandir  All Ages       Religious Temple     0.964525

    Mode_of_Transport
86           Bus,Taxi
9           Bus, Taxi
23          Bus, Taxi
89           Bus,Taxi
131          Bus,Taxi
```

[ ]:

```
```

[54]:

```
# User input
target_place = input("Enter the place you want to visit: ").strip()

# Check if it exists
if target_place not in df['Place_name'].values:
    print(f"'{target_place}' not found in the dataset.")
else:
    # Get target place details
    target_data = df[df['Place_name'] == target_place].iloc[0]
    target_cluster = target_data['Cluster']
    target_location = (target_data['Latitude'], target_data['Longitude'])

    # Recommend places in the same cluster, excluding the selected one
    recommendations = df[(df['Cluster'] == target_cluster) & (df['Place_name'] !
  ↪= target_place)].copy()
```

```
    # Calculate distances
    recommendations['Distance_km'] = recommendations.apply(
        lambda row: geodesic(target_location, (row['Latitude'],␣
↪row['Longitude'])).km,
        axis=1
    )

    # Suggest mode of transport
    def suggest_mode(dist):
        if dist < 1:
            return 'Walk'
        elif dist < 5:
            return 'Auto/Rickshaw'
        elif dist < 20:
            return 'Cab'
        else:
            return 'Metro/Bus'

    recommendations['Suggested_Transport'] = recommendations['Distance_km'].
↪apply(suggest_mode)

    # Sort by nearest
    recommendations = recommendations.sort_values('Distance_km')

    # Display top 5 recommendations
    print("\nTop nearby recommendations:")
    print(recommendations[['Place_name', 'Age', 'Category', 'Distance_km',␣
↪'Mode_of_Transport']].head())
```

```
Enter the place you want to visit:  National Library


Top nearby recommendations:
                    Place_name                        Age  \
121  Alipore Zoological Garden                   All Ages
24                 Alipore Zoo  Children, Teens, Families
149                 Taj Bengal                   All Ages
91         Alipore Jail Museum                   All Ages
32                 Chowringhee       18+ (for adult only)

                Category  Distance_km         Mode_of_Transport
121       The oldest zoo     0.474046            Bus,Taxi,Train
24            Zoo/Family     0.493581                 Bus, Taxi
149         Luxury Hotel     0.509808  Taxi, Cab, Auto-rickshaw
91      Historical place     1.092141                  Bus,Taxi
32   Commercial/Shopping     1.291664                 Bus, Taxi
```

[ ]:

```

```

[64]:

```
# User input
target_place = input("Enter the place you want to visit: ").strip()

# Check if it exists
if target_place not in df['Place_name'].values:
    print(f"'{target_place}' not found in the dataset.")
else:
    # Get target place details
    target_data = df[df['Place_name'] == target_place].iloc[0]
    target_cluster = target_data['Cluster']
    target_location = (target_data['Latitude'], target_data['Longitude'])

    # Recommend places in the same cluster, excluding the selected one
    recommendations = df[(df['Cluster'] == target_cluster) & (df['Place_name'] !
 ↪= target_place)].copy()

    # Calculate distances
    recommendations['Distance_km'] = recommendations.apply(
        lambda row: geodesic(target_location, (row['Latitude'],␣
 ↪row['Longitude'])).km,
        axis=1
    )

    # Suggest mode of transport
    def suggest_mode(dist):
        if dist < 1:
            return 'Walk'
        elif dist < 5:
            return 'Auto/Rickshaw'
        elif dist < 20:
            return 'Cab'
        else:
            return 'Metro/Bus'

    recommendations['Suggested_Transport'] = recommendations['Distance_km'].
 ↪apply(suggest_mode)

    # Sort by nearest
    recommendations = recommendations.sort_values('Distance_km')

    # Display top 5 recommendations
    print("\nTop nearby recommendations:")
    print(recommendations[['Place_name', 'Age', 'Category', 'Distance_km',␣
 ↪'Mode_of_Transport']].head())
```

```
Enter the place you want to visit:  Alipore Jail Museum


Top nearby recommendations:
                    Place_name       Age            Category  Distance_km  \
20             Kalighat Temple  All Ages  Religious/Spiritual     0.542166
118           National Library  All Ages  The largest library     1.092141
79                   Muktangan  All Ages             Cultural     1.198859
149                 Taj Bengal  All Ages        Luxury Hotel     1.551826
121  Alipore Zoological Garden  All Ages       The oldest zoo     1.565281

            Mode_of_Transport
20          Bus, Taxi, Walking
118                   Bus,Taxi
79                   Bus, Taxi
149  Taxi, Cab, Auto-rickshaw
121             Bus,Taxi,Train
```

[ ]:

[65]:

```
# --- Accuracy Evaluation using Precision@k ---

# Assume ratings   4.0 are good
def is_relevant(rating):
    return rating >= 4.0

# Add a relevance column
recommendations['Relevant'] = recommendations['Place_rating'].apply(is_relevant)

# Number of top-k recommendations (can change to 5 or more)
top_k = 5
top_k_recs = recommendations.head(top_k)

# Calculate precision@k
relevant_count = top_k_recs['Relevant'].sum()
precision_at_k = relevant_count / top_k

print(f"\n  Estimated Precision@{top_k}: {precision_at_k:.2f}")
```

```
 Estimated Precision@5: 0.40
```

[ ]:

[67]:

```
# User input
target_place = input("Enter the place you want to visit: ").strip()

# Check if it exists
```

```
if target_place not in df['Place_name'].values:
    print(f"'{target_place}' not found in the dataset.")
else:
    # Get target place details
    target_data = df[df['Place_name'] == target_place].iloc[0]
    target_cluster = target_data['Cluster']
    target_location = (target_data['Latitude'], target_data['Longitude'])

    # Recommend places in the same cluster, excluding the selected one
    recommendations = df[(df['Cluster'] == target_cluster) & (df['Place_name'] != target_place)].copy()

    # Calculate distances
    recommendations['Distance_km'] = recommendations.apply(
        lambda row: geodesic(target_location, (row['Latitude'], row['Longitude'])).km,
        axis=1
    )

    # Suggest mode of transport
    def suggest_mode(dist):
        if dist < 1:
            return 'Walk'
        elif dist < 5:
            return 'Auto/Rickshaw'
        elif dist < 20:
            return 'Cab'
        else:
            return 'Metro/Bus'

    recommendations['Suggested_Transport'] = recommendations['Distance_km'].apply(suggest_mode)

    # Sort by nearest
    recommendations = recommendations.sort_values('Distance_km')

    # Display top 5 recommendations
    print("\nTop nearby recommendations:")
    print(recommendations[['Place_name', 'Age', 'Category', 'Distance_km', 'Mode_of_Transport']].head())
```

```
Enter the place you want to visit:  Botanical Garden in Kolkata


Top nearby recommendations:
                                        Place_name       Age  \
3                                 Shalimar Station  All Ages
165   Sabuj Sathi Krirangan (Howrah Indoor Stadium)  All Ages
```

```
160                                    Nature Park   Adults
139                                   Fancy Market  All Ages
136                          Shri Jagannath Temple  All Ages

                                   Category  Distance_km  \
3    Transportation Hub (Railway Station)     2.719953
165          Indoor Stadium, Sports Venue     2.813631
160                           nature view     2.847313
139                      Market, Shopping     3.423767
136               Temple, Religious Site     3.752161

                     Mode_of_Transport
3                      Train, Bus, taxi
165  Bus, Taxi , Auto-rickshaw, Train
160                          Bus, Taxi
139                  Bus, Taxi , Metro
136          Bus, Taxi, Auto-rickshaw
```

[ ]:

```python

```

[ ]:

```python

```

[58]:

```python
from sklearn.metrics import precision_score
```

[63]:

```python
# --- Accuracy Evaluation using Precision@k ---

# Assume ratings   4.0 are good
def is_relevant(rating):
    return rating >= 4.0

# Add a relevance column
recommendations['Relevant'] = recommendations['Place_rating'].apply(is_relevant)

# Number of top-k recommendations (can change to 5 or more)
top_k = 5
top_k_recs = recommendations.head(top_k)

# Calculate precision@k
relevant_count = top_k_recs['Relevant'].sum()
precision_at_k = relevant_count / top_k

print(f"\n  Estimated Precision@{top_k}: {precision_at_k:.2f}")
```

```
 Estimated Precision@5: 0.60
```

[ ]:

```python

```

[ ]:

```

```

[ ]:

```

```

[70]:

```
# User input
target_place = input("Enter the place you want to visit: ").strip()

# Check if place exists
if target_place not in df['Place_name'].values:
    print(f"'{target_place}' not found in the dataset.")
else:
    # Get details of selected place
    target_data = df[df['Place_name'] == target_place].iloc[0]
    target_cluster = target_data['Cluster']
    target_location = (target_data['Latitude'], target_data['Longitude'])

    # Filter places in the same cluster
    recommendations = df[(df['Cluster'] == target_cluster) & (df['Place_name'] !
 ↪= target_place)].copy()

    # Calculate distances
    recommendations['Distance_km'] = recommendations.apply(
        lambda row: geodesic(target_location, (row['Latitude'],␣
 ↪row['Longitude'])).km,
        axis=1
    )

    # Suggest transport mode
    def suggest_mode(dist):
        if dist < 1:
            return 'Walk'
        elif dist < 5:
            return 'Auto/Rickshaw'
        elif dist < 20:
            return 'Cab'
        else:
            return 'Metro/Bus'

    recommendations['Suggested_Transport'] = recommendations['Distance_km'].
 ↪apply(suggest_mode)

    # Sort and select top 5
    top_k = 5
    top_k_recs = recommendations.sort_values('Distance_km').head(top_k).copy()

    # Mark relevance (rating >= 4.0)
    top_k_recs['Relevant'] = top_k_recs['Place_rating'] >= 4.0
    # Compute Precision@k
    relevant_count = top_k_recs['Relevant'].sum()
    precision_at_k = relevant_count / top_k

    # Final output DataFrame
    result_df = top_k_recs[['Place_name', 'Age', 'Category', 'Distance_km',␣
↪'Place_rating','Mode_of_Transport', 'Relevant']]
    result_df = result_df.rename(columns={
        'Place_name': 'Recommended Place',
        'Place_rating': 'Rating',
        'Relevant': 'Relevant (Rating   4.0)'
    })

    # Display results
    print("\n  Top Nearby Recommendations:\n")
    print(result_df.to_string(index=False))

    print(f"\n  Precision@{top_k}: {precision_at_k:.2f}")
```

```
Enter the place you want to visit:  Baghbazar Ghat


  Top Nearby Recommendations:

                   Recommended Place      Age                    Category
Distance_km  Rating          Mode_of_Transport  Relevant (Rating   4.0)
                           Kumartuli All Ages Traditional potters' quarter
0.550063     4.4                   Bus,Taxi                         True
                   Sovabazar Rajbari All Ages              Historic palace
1.020923     4.1             Bus,Taxi,Metro                         True
                 Shobhabajar Rajbari All Ages              Historical Site
1.023124     4.3                  Bus, Taxi                         True
Cossipore Gun & Shell Factory Museum All Ages              Historic palace
1.371327     3.9                   Bus,Taxi                        False
                    Bhootnath Mandir All Ages       Temple, Religious Site
1.767719     3.9 Bus, Taxi , Auto-rickshaw                         False

  Precision@5: 0.60
```

```
[ ]:
```

```
[56]: # User input
      target_place = input("Enter the place you want to visit: ").strip()

      # Check if it exists
      if target_place not in df['Place_name'].values:
```

```
    print(f"'{target_place}' not found in the dataset.")
else:
    # Get target place details
    target_data = df[df['Place_name'] == target_place].iloc[0]
    target_cluster = target_data['Cluster']
    target_location = (target_data['Latitude'], target_data['Longitude'])

    # Recommend places in the same cluster, excluding the selected one
    recommendations = df[(df['Cluster'] == target_cluster) & (df['Place_name'] !
 ↪= target_place)].copy()

    # Calculate distances
    recommendations['Distance_km'] = recommendations.apply(
        lambda row: geodesic(target_location, (row['Latitude'],␣
 ↪row['Longitude'])).km,
        axis=1
    )

    # Suggest mode of transport
    def suggest_mode(dist):
        if dist < 1:
            return 'Walk'
        elif dist < 5:
            return 'Auto/Rickshaw'
        elif dist < 20:
            return 'Cab'
        else:
            return 'Metro/Bus'

    recommendations['Suggested_Transport'] = recommendations['Distance_km'].
 ↪apply(suggest_mode)

    # Sort by nearest
    recommendations = recommendations.sort_values('Distance_km')

    # Display top 5 recommendations
    print("\nTop nearby recommendations:")
    print(recommendations[['Place_name', 'Age', 'Category', 'Distance_km',␣
 ↪'Mode_of_Transport']].head())
```

```
Enter the place you want to visit:  Baghbazar Ghat


Top nearby recommendations:
                               Place_name       Age  \
95                              Kumartuli  All Ages
98                      Sovabazar Rajbari  All Ages
21                    Shobhabajar Rajbari  All Ages
```

```
104  Cossipore Gun & Shell Factory Museum  All Ages
147                      Bhootnath Mandir  All Ages

                         Category  Distance_km          Mode_of_Transport
95   Traditional potters' quarter     0.550063                   Bus,Taxi
98                Historic palace     1.020923             Bus,Taxi,Metro
21               Historical Site      1.023124                  Bus, Taxi
104               Historic palace     1.371327                   Bus,Taxi
147        Temple, Religious Site     1.767719  Bus, Taxi , Auto-rickshaw
```

[ ]:

```
```

EXTRA MODIFIED CODE

[98]:

```python
import pandas as pd
from geopy.distance import geodesic
import ipywidgets as widgets
from IPython.display import display, clear_output

# Load your dataset here if not already loaded
# df = pd.read_csv('your_dataset.csv')

# Dropdown menu with all unique place names
place_dropdown = widgets.Dropdown(
    options=sorted(df['Place_name'].unique()),
    description='Choose a Place:',
    style={'description_width': 'initial'},
    layout=widgets.Layout(width='60%')
)

# Output display
output = widgets.Output()

# Function to trigger recommendations when place is selected
def recommend_from_dropdown(change):
    with output:
        clear_output(wait=True)
        target_place = place_dropdown.value

        if target_place not in df['Place_name'].values:
            print(f"'{target_place}' not found in the dataset.")
            return

        # Get selected place details
        target_data = df[df['Place_name'] == target_place].iloc[0]
        target_cluster = target_data['Cluster']
        target_location = (target_data['Latitude'], target_data['Longitude'])
        # Filter recommendations from the same cluster
        recommendations = df[(df['Cluster'] == target_cluster) &
                             (df['Place_name'] != target_place)].copy()

        # Distance calculation
        recommendations['Distance_km'] = recommendations.apply(
            lambda row: geodesic(target_location, (row['Latitude'],␣
↪row['Longitude'])).km,
            axis=1
        )

        # Transport suggestion
        def suggest_mode(dist):
            if dist < 1:
                return 'Walk'
            elif dist < 5:
                return 'Auto/Rickshaw'
            elif dist < 20:
                return 'Cab'
            else:
                return 'Metro/Bus'

        recommendations['Suggested_Transport'] = recommendations['Distance_km'].
↪apply(suggest_mode)

        # Top 5 recommendations
        top_k = 5
        top_k_recs = recommendations.sort_values('Distance_km').head(top_k).
↪copy()
        top_k_recs['Relevant'] = top_k_recs['Place_rating'] >= 4.0

        # Precision@k
        precision_at_k = top_k_recs['Relevant'].sum() / top_k

        # Final output DataFrame
        result_df = top_k_recs[['Place_name', 'Age', 'Category', 'Distance_km',
                                'Place_rating', 'Mode_of_Transport',␣
↪'Suggested_Transport', 'Relevant']]
        result_df = result_df.rename(columns={
            'Place_name': 'Recommended Place',
            'Place_rating': 'Rating',
            'Relevant': 'Relevant (Rating   4.0)'
        })

        print(f"\n  Recommendations based on: **{target_place}**\n")
        print(result_df.to_string(index=False))
        print(f"\n  Precision@{top_k}: {precision_at_k:.2f}")
```

```python

# Attach the event handler
place_dropdown.observe(recommend_from_dropdown, names='value')

# Display dropdown and results output
display(place_dropdown, output)
```

```
Dropdown(description='Choose a Place:', layout=Layout(width='60%'),␣
 ↪options=('Aakriti Art Gallery', "Abanindra…
```

```
Output()
```

[ ]:

[ ]:

[ ]:

[ ]: